

# $P^4QS$ : A Peer to Peer Privacy Preserving Query Service for Location-Based Mobile Applications

Meysam Ghaffari · Nasser Ghadiri ·  
Mohammad Hossein Manshaei · Mehran  
Sadeghi Lahijani

Received: date / Accepted: date

**Abstract** The location-based services provide an interesting combination of cyber and physical worlds. However, they can also threaten the users' privacy. Existing privacy preserving protocols require trusted nodes, with serious security and computational bottlenecks. In this paper, we propose a novel distributed anonymizing protocol based on peer-to-peer architecture. Each mobile node is responsible for anonymizing a specific zone. The mobile nodes collaborate in anonymizing their queries, without the need not get access to any information about each other. In the proposed protocol, each request will be sent with a randomly chosen ticket. The encrypted response produced by the server is sent to a particular mobile node (called broker node) over the network, based on the hash value of this ticket. The user will query the broker to get the response. All parts of the messages are encrypted except the fields required for the anonymizer and the broker. This will secure the packet exchange over the P2P network. The proposed protocol was implemented and tested successfully, and the experimental results showed that it could be deployed efficiently to achieve user privacy in location-based services.

**Keywords** LBS · Trusted Anonymizer Server · Peer-to-Peer Anonymizing.

## 1 Introduction

With the technological growth, especially in the Internet and communication devices, available services are increasing. A new class of services that have emerged with the development of the Internet and smartphones with location

---

Department of Electrical and Computer Engineering, Isfahan University of Technology, Isfahan 84156-83111, Iran. E-mail: {meysam.ghafari,m.sadeghi}@ec.iut.ac.ir {nghadiri,manshaei}@cc.iut.ac.ir

Corresponding Author: Nasser Ghadiri

sensors are location-based services (LBS). An LBS system provides one or more location-dependent services customized for every user that connects to the LBS provider and sends her location information to the server.

As LBS combines the cyber and physical worlds, it would be interesting for users. Example applications are finding nearby places such as restaurants and gas stations, or finding nearby friends through location-based friend finders, such as Foursquare.

Although such services are attractive, they need access to physical information about the real-world user's life [1]. If the system fails to provide adequate security mechanisms, it can be harmful to the users' privacy. For example, an intruder can infer the user's behavior, habits, social activities, sickness and other information just by knowing her location [2]. Therefore, the users' security and privacy must be regarded as one of the most important issues in LBS.

The privacy aspects of the location data have been extensively investigated. Most researchers acknowledge the need to hide the users' locations even from the service providers [3]. With the development of storage devices and the increased processing power of computers, the users could be easily identified by tracking their movements and the locations that they pass through. Furthermore, analyzing such trajectory data is possible [4], and most LBS users are not comfortable with this issue. It can cause many threats to the users' privacy. For example, users may be exposed to location advertisements, or subject to prejudice by service providers, or physical harm and extortion [2]. So, many researchers have proposed different methods such as spatial and temporal cloaking [5][6], fake locations [7], the addition of noise [8],  $k$ -anonymity [5], and similar methods to cope with this threat.

A well-known identity protection mechanism is  $k$ -anonymity [9]. The principle operation of this method is as follows: The system will preserve the privacy of a location-dependent query from a user if the probability of distinguishing the user's query is less than  $\frac{1}{k}$ . A trusted third-party anonymizer is used to achieve this goal [10][5][9] [11] (see Fig. 1). The trusted server interacts with users and manipulates their location data. Users will send their queries to the anonymizer instead of sending them directly to the LBS server. The anonymizer generates a cloaking region around the user in a way that the user is indistinguishable from  $k-1$  other users. Although this method seems to be working, it has its shortcomings. A serious challenge is to find a tradeoff between privacy, time and the quality of service.

A familiar privacy metric is based on the concept of anonymity, implying that the user is indistinguishable from other users as members of an anonymity set. Another privacy metric is unlinkability, which means the attacker cannot distinguish whether two or more items of interest have a relationship between them. In unlinkability approach, the information revealed by a sequence of queries is considered important because of the computation and storage characteristics. Most of the acquired information items are gained by analyzing the sequence of queries [12]. We will employ these metrics to preserve anonymity and unlinkability.

As discussed above, most of the existing methods are just a trade-off between the quality of service and privacy. Some of them suffer from the risk of revealing information in security bottlenecks, leading to sensitive information about the users to be disclosed. While the unlinkability factor is crucial, it has not been considered thoroughly in literature.

In this paper, a novel architecture named Peer to Peer Privacy Preserving Query Service ( $P^4QS$ ) is proposed for improving the users' privacy in location-based social networks. First, we use a distributed P2P architecture to acquire the queries and deliver the results. Second, by using proper symmetric and asymmetric encryption methods, each node would have access to the necessary information only. Third, by exploiting the Distributed Hash Table (DHT) in the proposed P2P architecture, the response is delivered to the client, but the LBS could not identify him. The main contributions of this paper are threefold:

1. We propose a novel P2P model for anonymizing the location-dependent queries. In this model, the nodes collaborate with each other to make the queries  $K$ -anonymized. At the same time, the information contents of the queries are protected.
2. The protocol has four advantages:
  - It does not require any trusted nodes. There are few distributed methods for anonymization, but they need a few trusted nodes to achieve anonymity.
  - Unlike existing distributed methods, the nodes do not need to store any information such as geographical maps.
  - The system has no security, privacy or computational bottleneck. For example in central trusted server, the server itself is the bottleneck of the service.
  - There is no additional or hidden costs for deploying the system. Unlike existing methods that would be expensive to implement due to their need for specific hardware and systems, the proposed method could be easily implemented with no prerequisites.
3. We have implemented the  $P^4QS$  as a proof-of-concept prototype to evaluate its feasibility. Full source code is made available for download<sup>1</sup>.

The rest of the paper is organized as follows: in Section 2, the related works will be surveyed. In Section 3, the basic concepts and a high-level view of the proposed model will be introduced. In Section 4, the proposed model and protocol description will be explained and evaluated in detail. In Section 5, both analytical and numerical evaluations of the protocol are illustrated by using a set of common scenarios and experiments. Finally, Section 6 closes the paper with the conclusion.

---

<sup>1</sup> <http://dkr.iut.ac.ir/content/codes-Peer-to-Peer-Privacy-Preserving-Query-Service>

## 2 Related Works

In this section, we give an overview of the current literature in location privacy domain. The techniques are categorized to the pseudonym, perturbation, and trusted server methods, as well as distributed architectures explained in the following subsections.

### 2.1 Pseudonym

Using the real identity has the risk of revealing information about users. The pseudonym method is proposed to address this problem. It uses a fictitious name instead of the user identifier [13]. However, it has been proved that by tracing the sequence of activities and queries, the identity of the user is detectable. The mix zone method further develops this core approach, which defines specific zones and allows the pseudonyms to change when the user changes his region [14] [15]. By changing the pseudonym, the unlinkability metric is satisfied whenever the user moves into another area.

### 2.2 Perturbation and Obfuscation

Another class of methods to achieve location privacy use perturbation and obfuscation. Adding noise, spatial cloaking, and fake location methods fall into this category.

The noise addition method interleaves noisy queries between the user queries to hide the original query. A problem with this approach is the high computational overhead on the server to process all queries [8]. The fake location method improves this approach by sending fake queries. Besides the real path of the user and his queries, a false path of queries will be generated and sent to hide the actual path of the user. However, the adversary could distinguish the authentic and fake paths by using background knowledge, map matching, or other methods such as using spatiotemporal density estimation and line intersections[8][16].

Spatial cloaking is a general method based on concealing the user queries by adding cloaks to achieve the  $k$ -anonymity. In this way, the user will become indistinguishable from  $k-1$  other users. In the primary method, a constant cloaking range is added to user queries. This method is optimized by proposing the trusted anonymizer server as explained in the following subsection.

### 2.3 Trusted Server

The methods above such as spatial cloaking, fake location, and noise addition put significant computational overhead on the server or imply the loss of quality. To overcome this problem, the use of trusted anonymizer server became popular, making a balance between privacy and the quality loss. The general

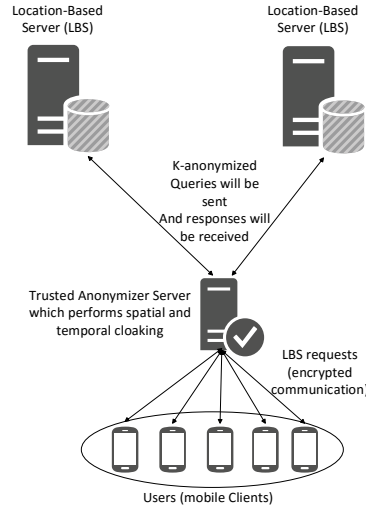


Fig. 1: Central Trusted Server architecture. In this architecture, all nodes send their requests to one trusted server. This server makes the queries  $K$ -anonymized and sends them to the LBS.

architecture of the trusted server is shown in Fig. 1. The basic idea is that each node sends its query to the trusted server. The trusted server accumulates the queries and anonymizes them by adding spatial and temporal cloaking in a way that at least  $k$  different queries become indistinguishable.

The trusted server architecture was first proposed by Gruteser and Grunwald [5]. It provides  $k$ -anonymity for protecting privacy based on a trusted anonymization server. They represent location information of the client by three tuples  $((x_1, y_1), (x_2, y_2), (t_1, t_2))$ . Tuples one and two show the area and the third tuple indicates the period of client queries. Based on this structure, they design a cloaking algorithm that generates spatiotemporal cloaking. The cloaked region contains, at least,  $k$  clients and their location information are sent to the location-based server indistinguishably. The parameter  $k$  is the minimum acceptable anonymity. In this method, it is very tough to provide a flexible privacy protection scheme, because of its pre-defined constant value  $k$ . Moreover, in the case of small areas,  $k$ -anonymity is very hard to achieve.

Significant efforts have been made to overcome these drawbacks and some novel methods are proposed, including CliqueCloak [10]. CliqueCloak is a personalized anonymity method by which clients can tune their personal privacy protection requirements as well as their spatiotemporal cloaking levels. This personal level is achieved by modeling the anonymization constraints to find the best satisfaction conditions. Another framework proposed for customized anonymity is Casper [11]. Casper has a profile for each user's privacy settings to fulfill the requirements of every individual user. It consists of a value that determines  $k$  for the minimum acceptable cloaking in  $k$ -anonymity. Casper

also uses an incomplete pyramid structure to keep the user information dynamically [17].

Another scheme based on trusted server is Cache Cloak [18]. This method caches LBS responses to use them for subsequent queries. Thus, it decreases the number of queries and the chance of adversary to infer from queries. Exploiting this method can increase privacy without the loss of accuracy. However, this system suffers from scalability problem. It cannot be used in a large scale situation because it needs huge amounts of memory to save the queries and responses. Moreover, in the worst case, if each client has different types of queries from different location-based servers, the trusted server needs to cache an enormous amount of information. Thus, it becomes the bottleneck of their proposed system.

Schlegel et al. proposed dynamic grid system (DGS) which requires a semi-trusted third party to preserve user privacy [19]. In this method, the user could define her privacy level. In DGS different privacy levels has not different communication costs for the user. However, this method also needs a central server.

In all derivations of central trusted anonymizer server, the queries must be sent to the central anonymizer. This limitation makes the anonymizer a bottleneck for the system. If an intruder penetrates to the trusted server, he could extract client's information. It has been proved that using pseudonyms or even spatial cloaking cannot solve this problem [20].

We propose a novel architecture based on distributed trusted server to address the problems mentioned above. In our proposed protocol, even an intruder can attack, only a limited information would be revealed. The proposed method relies on a peer-to-peer distributed architecture. Before presenting our solution, we review some literature in this category.

Table 1: Functions of each peer in the system.

Party	Function
$P_{C_i}$ : Client Peer $i$	Sends a query and waits for the response. The system must defend against correlation of queries and $P_{C_i}$ .
$P_{A_j}$ : Anonymizing Peer	Each user node determines a user's chosen location and is responsible for the anonymization of queries in the zone nearby. If sufficient anonymization is not achieved, this node cooperates with nearby $P_{A_j}$ s.
$P_{B_m}$ : Broker Peer	Receives the response packet based on $Hash(Ticket)$ and waits for $P_{C_i}$ to ask for his packet. Then $P_{B_m}$ sends the response packet to $P_{C_i}$ .

## 2.4 Mobile and Distributed Architectures

Central trusted servers had the potential of becoming the single point of failure, causing system failure. The central servers also have the risk of being compromised or controlled by an intruder, making them not reliable. Thus, deploying a secure trusted server that works smoothly in real-world situations would be extremely hard to achieve. To solve this problem, mobile-based methods have been proposed. The advantage of the mobile-based method for the users is that they could preserve privacy without relying on the central servers. In this approach, several groups of mobile users collaborate with each other to achieve the desired privacy.

CAP provides  $k$ -anonymity by maintaining road density using quadtree followed by VHC mapping and perturbation [21]. VHC mapping is used to project a two-dimensional geographic space into a one-dimensional space which preserves the homogeneity of every point. The adjacent points will also remain close to each other [22]. In this way CAP tries to reduce the storage and computational overheads required for mobile-based methods of storing the maps to achieve  $k$ -anonymity.

Hu and Xu proposed a method to generate cloaking boxes based on the strength of receiving a signal from the nearby devices [23].

Chen proposed the LISA method. LISA is based on  $m$ -unobservability and tries to prevent the attacker from attributing any particular location to the user. LISA also predicts the user movements by exploiting Kalman Filter that leads to increased privacy [24].

Our proposed protocol employs mobile-based methods that are used to preserve anonymity and unlinkability. Unlike the methods mentioned above, the privacy of user will be maintained without revealing and exchanging location information. Users communicate with each other, but they cannot get any information about other user's location or queries. Unlinkability metric is also satisfied by using ticket instead of user identification number, the pseudonym, or any other identifier.

## 3 Proposed Model

The proposed method which is called Peer to Peer Privacy Preserving Query Service ( $P^4QS$ ) follows a role-based, peer-to-peer architecture. As it is shown in Table 1, each node may have different roles at any time to perform one or more of three functionalities: (1) querying, (2) anonymizing queries, and (3) brokering responses.

From an overall view, the system operates as follows. In the first step, a node needs to send its query. We call this node "Client Peer" because it needs the service and plays the role of a client in a peer-to-peer architecture. We designate clients by  $P_{C_i}$ , where  $C \in \{1, 2, \dots, N\}$ . Each user is located in a zone where she sends the query. We represent these zones by  $Z_i$ , where  $i$  shows the sum of latitude and longitude of the user.

The  $P_{C_i}$ 's query will be sent to another peer responsible for anonymizing the specific zone which contains the query location. The anonymization is the second functionality of each peer. In this role, the "Anonymizing Peer" gets queries from different clients corresponding to a specific zone for which this Anonymizing Peer is in charge. We represent this peer by  $P_{A_j}$ . Note that there exist  $N$  Anonymizing Peers, and each peer is responsible for the zone  $Z_i$ , which is represented by  $P_A^{Z_i}$ .

$P_{A_j}$  anonymizes the queries from multiple  $P_{C_i}$ s by adding appropriate spatial and temporal cloaking to them to make them K-anonymized and sends them to the LBS. The LBS processes the queries and sends each of them to the specific peer responsible for that response. This peer, which is called *Broker Peer*, is represented by  $P_{B_m}$ , where  $B \in \{1, \dots, M\}$ . The Broker Peer gets the responses from the LBS and keeps them until the specific client asks its response from the broker. The broker then sends the corresponding answer to its owner, i.e., each  $P_C$  asks the response from  $P_{B_m}$  and  $P_{B_m}$  sends back the response to the  $P_{C_i}$ . The overall process is shown in Fig. 2.

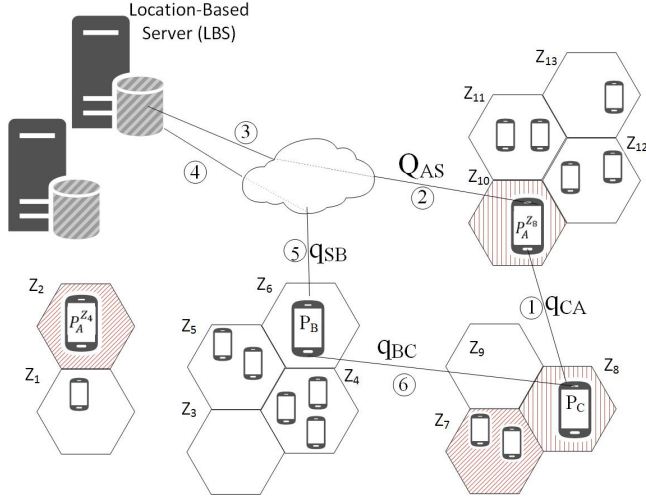


Fig. 2:  $P^4QS$  Architecture: Anonymizing and sending a query to the distributed system and getting the response. ① Client Peer sends query  $q_{CA}$  which includes Location, Proposed key, Ticket and Query. The proposed key is encrypted with the servers public key using RSA method. Ticket and query are encrypted with the proposed key using AES encryption method. ②,③ The Anonymizing Peer adds spatial and temporal cloaking to the queries and sends  $Q_{AS}$  set of K-anonymized queries to the LBS. ④ The LBS sends the encrypted (response)  $q_{SB}$  with the clients proposed key to the appropriate broker. ⑤ The broker will be chosen based on the calculated  $Hash(Ticket)$ . ⑥ The client interacts the broker and gets its response  $q_{BC}$ .



An essential point in the proposed architecture is that no central trusted server is required. Thus, the risk of the single point of failure will be eliminated. As shown in Algorithm 1, we first choose the nearest  $P_{A_j}$  based on the nearest neighbor of the query location (i.e., Line 3 in Algorithm 1). Achieving K-anonymity needs  $K$  different queries. So  $P_{A_j}$  needs to wait for queries from other users, or it has to generate fake queries. There is a wait time loop (line 4) such that if  $P_{A_j}$  does not receive enough queries (line 5), it will collaborate with the adjacent nodes or generate fake queries, as shown in line 6 of the algorithm. Then  $K$ -anonymized queries are sent to the server. The server will process the queries and send the encrypted response packet to  $P_{B_m}$  (i.e., line 11).  $P_{B_m}$  is chosen based on the *Hash (Ticket)*, where the ticket is sent by Client Peer in the first step. Finally,  $P_{C_i}$  asks  $R$  from  $P_{B_m}$  and  $P_{B_m}$  sends it to  $P_{C_i}$ .

So the overall  $P^4QS$  protocol procedure is as follows:  $P_{C_i}$  finds the appropriate  $P_{A_j}$  and also finds  $P_{B_m}$  based on  $\text{Hash}(\text{Ticket})$ . Then  $P_{C_i}$  sends its query to the  $P_{A_j}$  and request for the response of this query to the  $P_{B_m}$ . The  $P_{A_j}$  accumulates and anonymizes the queries. If needed,  $P_{A_j}$  adds fake queries to achieve the predefined privacy threshold. Then it sends these anonymized queries to the server. The server gets the queries, decrypts and processes each query, and sends the response to the node  $P_{B_m}$  based on  $\text{Hash}(\text{Ticket})$ . After receiving a response from server or a request for response from a client,  $P_{B_m}$  matches the request and response and forwards the answer to the client. The overall process was shown in Algorithm 1. The user's query is in the form of:

$$\text{Loc}, E(\text{prop-key})_{\text{Pub}_{\text{server}}}, E(\text{Ticket}, \text{Query}, P'_{B_m} \text{ sIP})_{\text{prop-key}}$$

where Loc is the location of the user and the query is based on this place.

1. The random number which is generated by the LBS to determine  $P_{B_m}$ .
2. A validation time stamp which shows the expiry time of the ticket.
3. The server mark which is encrypted with the private key of the server. Every user can decrypt it with the public key of the server to assure the validity of the ticket, in the case of receiving a ticket from other peers.

*prop-key* is the user's proposed key for the encryption of the response. This key can be changed for each request.  $\text{Pub}_{\text{server}}$  is the server's public key and is known for all users.

To evaluate the efficiency of the proposed model, we use a set of common challenging scenarios to assess this model from multiple aspects. These scenarios have different frequency of queries, time delay sensitivity, position accuracy and the amount of revealed information per query. Consequently, each scenario can evaluate the model from a different aspects. The features of these scenarios and their attributes are summarized in Table 2. We will explain the scenarios and evaluate the model by them in the evaluation section.

#### 4 Protocol description

The  $P^4QS$  is based on attaining two important goals: distributed anonymizing and avoid correlation between the sequences of queries (unlinkability). To

**Algorithm 1** The Pseudo code of the  $P^4QS$ 


---

```

1: Input: ClientQuery :
   Encrypted with proposed key(Ticket, Query) and Encrypted with servers public
   key(proposed key)
2: Output: Response of the query which will be delivered to Client with a distributed
   hash table
3:  $P_{C_i}$  Sends Query to  $P_{A_j}$ 
4: while Anonymity in  $P_{A_j} < K$  do
5:   Wait for new query
6:   Or
7:   Collaborate with  $P_{A_j} \pm 1$  (Adjacent  $P_{A_j}$ ) or generate fake queries
8: end while
9:  $P_{A_j}$  sends ( $K$  Anonymized Query) to the server
10: Server process queries
11: Server sends  $E(R)_{ProposedKey}$  to  $P_{B_m}$  which  $P_{B_m} = Hash(Ticket)$ 
12:  $P_{C_i}$  Sends Query to  $P_{A_j}$ 
13:  $P_{B_m}$  Sends  $R$  to  $P_{C_i}$ 

```

---

Table 2: Proposed scenarios and their usages

Scenario	Main feature	Proven feature by Scenario
<i>Profile Matching (PM)</i> : used for finding similar profiles	Accuracy is important but sequence of queries must not correlated	preserving accuracy, simultaneously no correlation between queries
<i>Driving Condition Monitoring (DCM)</i> : in case of need to monitor the road condition, privacy of the users who sent the data must be protected	low sensitivity to the position accuracy and delay but sequence of queries must be kept uncorrelated	correlation between queries in case of not sensitivity to delay and position accuracy
<i>Road Map (RM)</i> : used to find the path	Sensitive to delay but Position cloaking is tolerable	ability to protect against path tracking
<i>Detecting User By Sequence of Queries (DUSQ)</i> : can be used to detect user and visited places	Revealing user's identity by correlating queries to user	inability to associate queries to user
<i>Detecting User Speed in Highway (DUSW)</i> : even without detecting the user identity, detecting over-speeding in highways can jeopardize privacy	inability to correlate sequence of queries to each other	detecting correlation between queries

achieve these goals, we proposed a distributed peer-to-peer architecture. In the proposed system, each peer has three roles, as described in the previous section, namely *client*, *anonymizer*, and *broker* roles.

In the first step, each peer needs to send its query, but, unlike the previous methods, the query will be forwarded to the relevant anonymizer peer. Each peer in our system is responsible for anonymizing queries close to a particular location. The anonymizer peer gets multiple queries from multiple client peers. These queries have an important part in common, and that is their positions.

These locations of queries are close to each other and  $P_{A_j}$  anonymizes them and sends them to the LBS.

The structure of query is slightly different from typical queries. The query consists of the user's location which is sent as plain text, the ticket, the query text and the proposed key. The proposed key is encrypted with the public key of the LBS and the ticket and the query encrypted with the proposed key. The ticket is used as a control mechanism in our protocol and will be described later in this section. The queries are encrypted with the user's proposed key. For peer-to-peer communications, we exploit a distributed hash table (DHT) as described shortly. Then we will explain the proposed method and show how it uses the DHT.

#### 4.1 Distributed Hash Table (DHT)

Distributed hash table belongs to a class of decentralized systems for distributing tasks between peers. In this method, each peer is responsible for keeping data and forwarding it to others. The DHT architectures provide three major features: scalability, flexibility, and immediate deployment. It could be used efficiently in mobile nodes [25]. For finding a specific node in the DHT, different methods have been proposed. One of the best and most simple methods is Chord. In Chord, each peer stores the addresses of  $n$  peers. The computational complexity of the search operation is  $\log(N)$ , where  $N$  is the number of all peers in the distributed table [26]. As depicted in Fig. 3, if a peer has data and wants to locate or save this data, it will calculate the hash of data. The nearest peer to the extracted value is responsible for keeping or locating the data. For example, in Fig. 3, peer X is responsible for data "D" as its hashed value is the closest to the hash of data. Most of the proposed DHT use SHA-1 hash function.

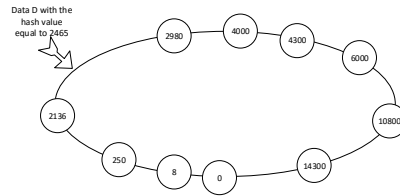


Fig. 3: DHT Search Space: each node is specified by an index number from 0 to  $2^n - 1$ . For finding a specific node, the user finds the closest node to the  $Hash(data)$ . In this case, there exist 10 active peers. The hash value of  $D$  is equal to 2465. Hence, peer 2136 is responsible for it.

## 4.2 Hashing Protocol

One of the advantages of DHTs is that the user could specify the appropriate hash function. In the proposed system, the protocol uses geographical data for this purpose, which consists of latitude and longitude. A straightforward and efficient method for using these two-dimensional data is to map it into a new one-dimensional space. For this purpose, we define our hashing function by,

$$H = X + Y, \quad (1)$$

where  $X$  and  $Y$  are the longitude and latitude values of the determined location of  $P_{C_i}$  and  $H$  is the hash value used in DHT for finding a particular peer. Using this function, we can simply find the best anonymizer peer for each location. Adjacent geographical anonymizer peers (peers whose RAs are close to each other) are located beside each other in the DHT, so they can easily cooperate with each other to achieve the best anonymization by optimum spatial and temporal cloaking.

The proposed function has one weakness in the first glance: the symmetric points will be matched to one point in the new space, as shown in Fig. 4.

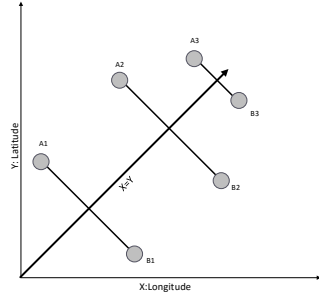


Fig. 4: The symmetric points. If some nodes locate in each of these symmetric locations, they will be assigned to one anonymizer based on the sum of their latitude and longitude.

This problem could be addressed by assigning two symmetric zones to each anonymizer. Thus, each anonymizer is responsible for both regions. After receiving the queries, the anonymizer separates them based on a defined threshold and anonymizes each zone separately.

## 4.3 Ticket

A key architectural element in our proposed method builds on the concept of the ticket. It will be used for running the  $P^4QS$  protocol by different peers. Unlike pseudonym-based methods, exchanging the ticket does not need any

computation or authentication. Therefore, it could be easily implemented and quickly used. Moreover, analyzing linkability between queries is impossible. The usage of the ticket is described in this subsection.

When the location-based server processes the query and produces the response, it needs to send it back to the client. The aim of our protocol is to make the client completely anonymous and eliminate the linkability of the queries to other queries or users. An identifier is needed to identify the owner of each query and simultaneously change it abruptly with the minimum computation cost. For this purpose, we propose using a unique ticket for each query. By this method, the owner of the query is unknown to the LBS and the only available information is the ticket, which is unique for each query. Thus, analyzing a sequence of queries becomes impossible.

The server(LBS) sends the response based on the ticket. A simple method for the LBS is extracting the hash of the ticket and sending the response to the appropriate peer. The peer with the closest hash to the hash of the ticket, which is called *Broker Peer*, is responsible for receiving the response from the server (LBS). In this case, the same hash function could be used in this step as well, and there is no need to define a new distributed hash table.

The *Client Peer* knows the ticket and thus, it can calculate the hash of the ticket and find out the appropriate *Broker Peer* to ask for the response.

#### 4.4 Protocol Execution

The  $P^4QS$  protocol executes in two phases: initialization and execution. In the initialization phase, the peers introduce themselves to the system. In the initialization process, each peer runs a server thread and listens to a particular port. Then it chooses *myRA*, which is a random number used for determining the anonymizer and the broker.

After receiving the response to the join request, the client is ready to perform all his functions as  $P_{C_i}$ ,  $P_{A_j}$  and  $P_{B_m}$ . The pseudo codes for these functions are shown in Algorithm 2, Algorithm 3, and Algorithm 4, respectively.

---

#### Algorithm 2 Pseudo code of $P_{C_i}$ (*Client Peer*)

---

- 1: *Get the Location*
  - 2: *Choose myRA*
  - 3: *Calculate Hash = X + Y of the Location*
  - 4: *Choose one of the tickets and retrieve its Hash bydecrypting the ticket with Servers public key*
  - 5: *Find Anonymizer*
  - 6: *Find Broker (Based on ticket)*
  - 7: *Send Query : Loc, E(PropKey)<sub>PubServer</sub>, E(Ticket, query, Broker'sIPaddress)<sub>PropKey</sub>*
  - 8: *Send "Request – For – Answer" message to Broker*
-

As shown in Algorithm 2,  $P_{C_i}$  finds the appropriate anonymizer based on  $Hash(location)$  and sends the query to it. Also, the broker is chosen based on the  $Hash(Ticket)$ . So, by changing the users' location, the anonymizer peer will be changed. Moreover, each response is sent to a broker. Thus, compromising a client (a broker peer) by the server will not reveal significant information about a specific user.

---

**Algorithm 3** The Pseudo code of  $P_{A_j}$  (*Anonymizing Peer*)

---

```

1: while Anonymizer Peer is Active do
2:   while Anonymity in  $P_{A_j} < K$  do
3:     if New Query Recieved then
4:       Put query in ReceivedQueries list
5:     end if
6:     Wait  $T$  second for new queries
7:     if after  $T$  second Queries  $< K$  then
8:       Collaborate with  $P_{A_j} \pm 1$  (Adjacent  $P_{A_j}$ )
9:     OR
10:    Generate Fake Query(ies)
11:   end if
12: end while
13:  $P_{A_j}$  sends ( $K$  Anonymized Queries) to the server
14: Empty Query List
15: end while

```

---

Algorithm 3 shows the function of the anonymizer peer. In this procedure, the anonymizer is always waiting for new queries. In lines 3 and 4, after a new query is received, the anonymizer puts it in the query list. Lines 6-11 indicate that the anonymizer waits for  $T$  seconds to gather  $K$  queries. If the number of received queries is less than  $K$ , she can collaborate with the adjacent anonymizers or create some queries herself. Then, the anonymizer adds spatial cloaking to the  $K$  queries to make them indistinguishable. Since the ticket information is encrypted by the public key of the LBS, the anonymizer has no access to the sensitive information of the user, so she just could anonymize the user location. Thus even in case of having a malicious or semi-honest anonymizer she just has access to the user location in a specific zone. As soon as the user moves to an adjacent zone, she will be assigned to another anonymizer so the anonymizer could not analyze the user path.

Functioning as a broker peer, the client saves requests for the answers received from other clients and the responses received from the LBS in two separate lists. As soon as a new request for answer or response is received, it checks the list and if it finds a match, it will forward the response to the  $P_{C_i}$  and remove both request and response from the lists (as shown in Algorithm 4).

As described in Section 4.3, the system uses tickets to deliver responses to the owner. Although using multiple pseudonyms is an ideal method to achieve privacy, it has the risk of the (Denial of Service) DoS attack, leading to system breakdown [27]. The  $P^4QS$  is similar to multiple pseudonyms method, but it does not require the pseudonym changing process, eliminating the need

**Algorithm 4** The Pseudo code of  $P_{B_m}$  (*Broker Peer*)

---

```

1: while Broker Peer is Active do
2:   Wait for new message
3:   if New Response(Request for Response) Received then
4:     if There is a Match for it in Request List(Response List) then
5:       Remove the Matched Request (Response) from the Request List (Response List)
6:     end if
7:   else
8:     PUT the Request (Response) in the Request List (Response List)
9:   end if
10: end while

```

---

for time and computational process. The  $P^4QS$  could be vulnerable to DoS attacks especially because, unlike multiple pseudonyms methods, it does not have any authentication. So, the proposed method is ideal for privacy, because it uses tickets that work like a pseudonym for each query. However, the protocol needs to be protected against DoS attacks. To solve this problem, we propose a ticket exchange protocol.

If an intruder wants to make a DoS attack on the system, he will need to send plenty of valid queries. We prevent such attacks by limiting the allowed number of queries of each user. To achieve this goal, we propose a ticket exchange protocol. In this protocol, the location-based server generates a collection of tickets and sends a specific number of tickets to each authorized user of the system in predefined time periods; for example, in an hourly manner. The client uses one of these tickets for each query. Thus, the DoS attack becomes preventable. However, this ticket exchange approach has two issues: using randomly generated tickets by the attacker and tracking the users by knowing the owner of the tickets. To solve these problems, we further improve the ticket exchange protocol.

**Ticket Generation:**

The server will generate a pre-defined number of tickets and send them to the users in a specific time periods. The structure of ticket sent by the LBS is  $E(Token, Identifier, ValidTime)_{Priv_{Server}}$ , which means the ticket is an encrypted package including a random token, an identifier and the valid time of the ticket. This package is encrypted with the private key of the LBS. So each entity in the system can decrypt it with the public key of the LBS and check the identifier to control the validity of the package. This mechanism has two important roles in the system. First, the attacker could not generate random tickets and the LBS checks the identifier. Before processing the query, the LBS checks the validity of the ticket number and thus, the chance of the intruder will be lowered significantly. The second and more important role of this identifier is that in the  $P^4QS$ , users exchange some of their tickets with each other in defined time intervals frequently. The tickets are chosen randomly for exchange and each user exchanges tickets just with the adjacent peers. The user checks the identifier after getting each ticket and thus, she can validate the tickets. If any user gets any invalid tickets, she drops them. If the

invalid tickets exceed a predefined threshold, she will consider the sender as untrusted, so generating and distributing fake tickets will be prevented.

After a few exchanges, the tickets will be circulated across the network of the users, and the LBS may not be able to determine the owner of any query. Thus, we prevent DoS attack and provide the ideal privacy. The pseudo code for the ticket exchange protocol is shown with Algorithm 5. In this algorithm,  $P_{C_{t_i}}$  is a list of trusted peers for peer  $P_{C_i}$  and  $P_{C_{t_k}}$  is a peer in this list.

---

**Algorithm 5** Ticket Exchange Protocol

---

```

1: Ticket:  $(token, verifier, TimeStamp)_{private_{server}}$ 
2: Server sends ticket batch to each authenticated user in specific time periods
3: while not achieved ideally exchanged tickets do
4:    $P_{C_i}$  exchange  $E$  of its randomly selected Tickets with other  $P_{C_j}$ 
5:   Check the validity of each received ticket
6:   if Received ticket is invalid then
7:     Drop the ticket
8:     count number of invalid tickets from specific  $P_{C_j}$ 
9:     if  $P_{C_j}$  sends invalid tickets more than threshold then
10:      Remove  $P_{C_{t_k}}$  from trusted list
11:     end if
12:   end if
13: end while
14: Each peer has limited valid tickets that the server could not trace them

```

---

In our ticket exchange protocol, if the LBS has  $T$  tickets for each peer in the defined periods, and the system has  $N$  peers, every peer will exchange  $E$  tickets in each round. The exchange could be done by sending  $E$  tickets for the following peer. So after  $R$  rounds, the probability of having tickets assigned by the LBS to each peer can be calculated by Equation (2).

$$P = \left(\frac{T-E}{T}\right)^R + \left(\frac{E}{T}\right)^N \sum_{i=1}^{R-1} \left(\frac{T-E}{T}\right)^i \quad (2)$$

If the number of peers gets high enough, as compared to the number of tickets, The sigma part of the equation could be neglected. Achieving low  $P$  is essential for the privacy of users (preferably  $P < \frac{1}{K}$ ).

#### 4.5 Handling Overloading Anonymizer

Every system faces inappropriate situations. Most of the system failures and attacks are due to these problems. In this case, discovering the weaknesses and handling the problem is critical. Based on our analysis, in the case of using the  $P^4QS$ , the overloading anonymizer exceptions need to be handled properly. As described in the protocol description method, each anonymizer peer is responsible for anonymizing specific zones. As these zones are randomly selected for each peer, it is possible to have an anonymizer in a crowded zone.



In this case, it is possible to have too many queries. The anonymizer is just a mobile node, and it is unable to handle all queries. In this case, a right solution is to allow this node to leave the DHT and rejoin with a new random *MyRA*. The anonymizer is now responsible for the new zones. At the same time, when the anonymizer leaves the DHT, the queries will be sent to the closest peers, which means that the queries will be divided between two peers, the predecessor and successor nodes.

## 5 Evaluation

In this section, we evaluate the  $P^4QS$  by providing common scenarios in Section 5.1 and this is followed by evaluation against the scenarios in Section 5.2. A proof-of-concept prototype of the system is also built and evaluated in Section 5.4.

### 5.1 Scenarios

We illustrate the different aspects of the proposed method through a few scenarios. These services are different from the following aspects:

- *The frequency of access*: This defines the acceptable period between two sequential queries.
- *Time accuracy and delay sensitivity*: The adequate temporal cloaking which may be caused by the anonymizer due to network delay.
- *Position accuracy*: The acceptable location cloaking which is made by the anonymizer.
- *The amount of required (revealed) data*: In the case of requesting a specific service, what information must be sent (revealed) by the user?

#### 5.1.1 Profile Matching

One of the possible applications of location services could be profile matching. In this case, a user needs to know specific information about users nearby. It is very important just to reveal the permitted information, and the user should not be able to perform further analysis by sending multiple queries. In this case, accuracy is critical, but a sequence of queries or responses must not be correlated.

#### 5.1.2 Driving Conditions Monitoring

Modern vehicles have multiple sensors to acquire road and weather conditions. These sensors could be useful for reporting road conditions instead of using multiple expensive fixed sensors. The vehicle will report road condition alongside its path, and the whole road will be covered with sensors. In this case, the position accuracy is not so critical, and a distance of 100 meters will be

sufficient [5]. This situation is not sensitive to delay, and reporting on the condition with some delay is tolerable. However, the method must prevent the extraction of further information such as path and speed from the reported conditions [28][29].

### 5.1.3 Road Map

The roadmap is a useful feature in routing. In this case, the user needs to report his position and query the map and specific points of interest such as hospitals, hotels, and so on. In this case, a quick and accurate response is needed, but the query could have some spatial cloaking because the answer will cover a large area around the query. The important issue here is that the identity or goal of the user must not be revealed from sequences of queries or a particular location.

### 5.1.4 Detecting User based on the Sequence of Queries

It is important that the intruder must not be able to reveal the user or his goal by analyzing sequences of queries, regardless of the type of service or user's intention. In this case, two important issues must be noticed. First, there is no correlation between queries or between the queries and users. Second, the exact position of the user in any query must not be revealed, because in this case, it is probable that intruder could identify the user and his goal is based on the query and background knowledge. For example, if a user queries hospitals in midnight, when he is at his home, the intruder finds that the specific person is sick, so she is going to reach a hospital.

### 5.1.5 Detecting User Speed in Highways

Sometimes just knowing the sequence of queries even without knowing the user could be harmful to the user. For example, the police or insurance companies could detect over-speeding in highways and detect the user in cameras based on his position. In this case, the location-based service is not desired, and the user prefers not to use it. Thus spatial cloaking or time delay is not a solution here, and the only possible solution is that there would be no way to correlate the sequence of queries. By doing that, it is probable that next query could be from another user, and the privacy of the user might be assured.

## 5.2 Scenarios Evaluation

In this section, we are going to analyze the  $P^4QS$ . First, we use the mentioned scenarios to evaluate and analyze it thoroughly from different aspects.

The most frequent scenarios could be driving condition monitoring and road maps. In these scenarios, the accuracy and delay are ignorable. The aim is to hide the users' identity in these scenarios, as it could be achieved in most

of the previous methods such as fake locations, noisy queries, trusted server, and so on. The same goal could be obtained with the  $P^4QS$  because the user's query will be anonymized by spatial and temporal cloaking in the anonymizer peers.

In the scenario of detecting user by a sequence of queries, the intruder needs to know the owner of multiple queries. This is possible in the previous methods such as pseudonyms. In the  $P^4QS$ , each query has its unique identifier, which is the ticket, and thus the attacker will not be able to analyze the sequence of queries. This situation happens for the profile matching scenario by which the attacker knows the user and wants to distinguish it from other users, thus tracing his activities. In this kind of attack, using fake paths does not solve the problem because the attacker can easily distinguish them by knowing the exact pattern of the user. But the proposed model is resistant against this attack also by using tickets. In these cases, even without considering spatial cloaking (which is not recommended), the attacker cannot trace the user and distinguish the sequence of user's queries and hence, his identity, path and activities. The proposed method satisfies the unlinkability metric and thus preserves the privacy of the user. Note that in the central trusted server approach, the system is vulnerable to these threats in the case of compromising the trusted server or gaining control over it by the intruder.

The final scenario is detecting user's speed in highways. Most of the previous methods do not preserve the user against this threat because they do not satisfy the unlinkability metric. Even knowing two consecutive queries is sufficient to detect over-speeding. In this case, by using cameras, the user is distinguishable by knowing the approximate location and using the pseudonym, spatial cloaking, and the fake path could not help. Since the only required information is two correlated queries, by using other methods, the user will be distinguished. A central trusted server is not completely trustable because it could be controlled by the government or the insurance companies. The proposed method is reliable in this case because in this model, there are not any correlated queries, and no one could determine that two queries belong to one user. Thus, over-speeding detection is impossible.

### 5.3 Attacks against $P^4QS$

In this part, we are going to explain the possible attacks against the  $P^4QS$ . It uses tickets and satisfies anonymity and unlinkability metrics and thus preserves user's privacy. However, using tickets will cause the vulnerability in some situations as explained below.

**Non-cooperative nodes:** It is possible that  $P_{A_j}$  or  $P_{B_m}$  may not cooperate with the users, and the user could not get its response during this period. In this case, it would be hard for the user to find the non-cooperative peer since he only knows that the response is not available. If the problem is with the  $P_{B_m}$ , it can be easily solved by sending another query. By having a new hash value, the new broker will be assigned to deliver the response. However,

since the user is in a particular location, the  $P_{A_j}$  is constant and thus, the system needs to determine the non-cooperative peers and remove them from the hash table. This could be done by announcing the non-delivery of the response. If these announcements exceed a specific threshold, the  $P_{A_j}$  will be removed from DHT, but this threat remains in the proposed model.

**Sending used tickets to other peers:** In the  $P^4QS$ , each peer exchanges its tickets with other trusted peers to avoid identification by the LBS through using specific tickets. In this case, a peer can have malicious activities such as sending invalid tickets or used tickets. Invalid tickets could be recognized through the identifiers as mentioned in Section 4.4, but the used tickets cannot be determined. The only possible solution is that the server adds a time stamp to the tickets to show their valid time. Thus by limiting the usage time of the tickets, a malicious peer has does not have enough time to use and resend the tickets to other peers, but the model is still vulnerable against this attack.

#### 5.4 Experimental Evaluation

The  $P^4QS$  is evaluated by implementing a server-based system and a proof-of-concept application on Android operating system to prove the feasibility and evaluate the overall performance of the protocol. For this purpose, two experiments are performed: (i) We first perform some tests on a local network (Android device, emulator, and server are all connected to a local, isolated network), (ii) then the same tests are performed in the Internet.

In the implemented protocol, each user chooses a random pair in  $(-270, 270)$  interval, that is called  $RA$ . Note that the sum of each pair of (longitude, latitude) on the Earth is in this interval. Clients, either as real Android devices or virtual clients in the emulator, will form a DHT in the server and will be sorted based on their  $RA$ s. Also, each client has the pair  $(RA, IP \text{ Address})$  of four successors and four predecessors of itself in the DHT. The  $N^{th}$  successor of client  $A$  is  $2^{(N-1)}$ th node in the DHT after  $A$  and  $M^{th}$  predecessor of client  $A$  is  $2^{(M-1)}$ th node in the DHT before  $A$ .

The implemented querying process consists of a few steps explained below: First, the client needs to find its anonymizer. So he calculates the sum of its longitude and latitude and then finds the client whose  $RA$  is closest to the calculated sum. After that, the client chooses one of its tickets to put it in the query and find the broker as well.

Tickets are created by the server and sent directly to clients (for example, in reply to a join message). For creating a ticket, the server chooses a random pair in  $(-270, 270)$  interval and attaches it to the known string "server" and finally, encrypts it with the server's private key. So, it is only the server that can create valid tickets.

After the client chooses a ticket, it decrypts the ticket with server's public key. Now, the client can find the broker of the query. The broker is the client with the nearest  $RA$  to the random number of the tickets. Now it is the time to create the query. Query is a message that consists of four strings:

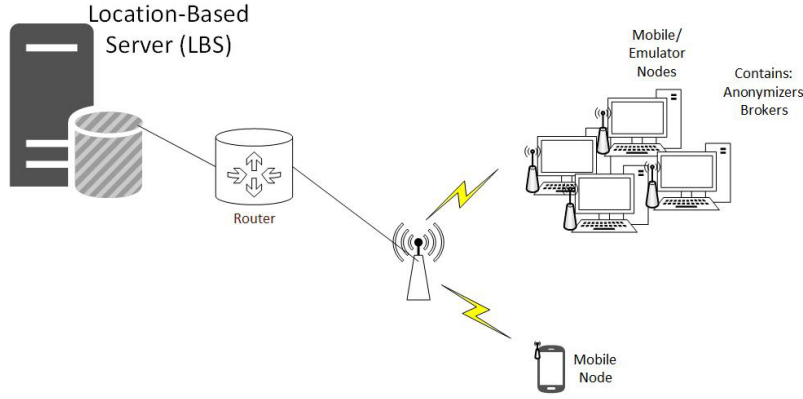


Fig. 5: Local Network Architecture

- 1- Client's longitude
- 2- Client's latitude
- 3- (Query text + ticket + broker's IP address) encrypted with a symmetric key.
- 4- (The symmetric key) encrypted with the server's public key.

Then the query is sent to the anonymizer, and a request for the answer to this query is sent to the broker. The anonymizer waits for five seconds (or less if four queries have been received) to gather some queries. If there are less than four queries received, the anonymizer will create some queries to have four queries to anonymize. Now, the anonymizer changes the longitude and latitude of the queries to anonymize them. This process is a trade-off between the quality of service (accuracy of the answer) and privacy. The anonymized queries are sent to the server. The server decrypts the ticket of each query, checks the validity of the ticket and creates an answer message, and sends it to the broker responsible for that answer. When the broker receives an answer from the server, it decrypts the ticket, checks the validity of the ticket and then sends the answer back to the client who has requested for that answer.

#### 5.4.1 Local Evaluation

In this experiment, the server was connected to an access point using an ethernet cable and the cell phone and emulator nodes were attached to the same access point using a wireless connection. (As shown in Fig. 5 )

Hardware specifications of devices used in this experiment are as follows:

**Android device:** Samsung GT-I9000 mobile phone, which has 1GHz Cortex-A8 CPU, 512MB of RAM, 8GB of internal memory and Android version 4.4.4.

**Emulator:** A 2.6GHz Dual-core Intel Core i5 computer, with 8GB RAM and 256GB PCIe-based flash storage.

**Server:** A desktop computer with Dual-Core CPU E5300 @ 2.60GHz, x64-based processor and 2GB RAM.

In this experiment, two different approaches were taken. First, we ran Android application and the emulator as the ordinary clients. The Android device and virtual devices were used to send queries and receive the answer.

Table 3 shows the mean of the querying time (the time between sending the query to the anonymizer and receiving the answer from the broker) and the mean time of finding the broker for the mobile node and emulator nodes. Wait time is a random time the client waits after receiving an answer. After that time, the client sends another query.

Table 3: Wait time for combination mode within local network

<b>Time(ms) / Number of Nodes</b>	<b>20</b>	<b>40</b>	<b>60</b>	<b>80</b>
<i>Receiving query mean time (emulator)</i>	4855.44	4993.72	5071.60	5802.11
<i>Finding broker mean time (emulator)</i>	10.98	14.03	9.76	5.21
<i>Receiving query mean time (mobile)</i>	4354.41	4867.50	5199.06	5361.60
<i>Finding broker mean time (mobile)</i>	46.48	31.63	45.73	53.71

Moreover, a particular configuration was tested to analyze the abnormal situations. In this test, the anonymizer of all clients in the emulator is set to be the real device. So the load effect of the protocol on a single node is analyzed. Table 4 shows the results of this test.

Table 4: Sending all load on a single mobile node in local network

<b>Maximum wait time (ms)/number of emulator nodes</b>	<b>20</b>	<b>50</b>	<b>100</b>
<i>10000</i>	678.42	1668.78	2947.87
<i>20000</i>	1492.32	1585.00	2712.63
<i>30000</i>	1828.21	2272.16	2332.19
<i>40000</i>	2156.14	2319.00	3122.87
<i>50000</i>	2062.66	2486.10	3042.99
<i>60000</i>	1954.89	3267.05	3019.08

As the results indicate, the processing load of the proposed method is not significant for any user node over the network.

#### 5.4.2 Internet-based Evaluation

In this part, the protocol was evaluated over the Internet using multiple devices with different connection types. The cell phone was connected to the Internet using a wireless ADSL modem. For the emulator and the server, two virtual private servers with valid IP addresses were used. So this test could be more close to the real-world infrastructures. (Fig. 6)

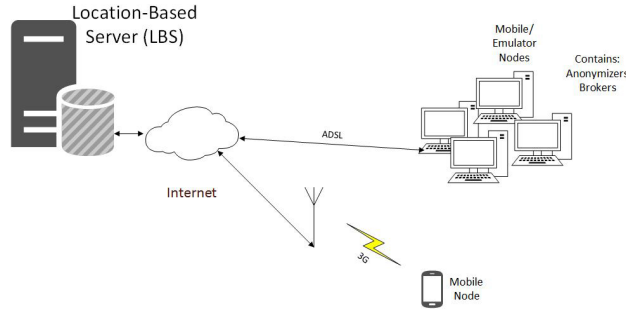


Fig. 6: Experimental architecture over the Internet

Table 5 shows the mean of querying time over the Internet with different wait times before sending the next query.

Table 5: Wait time for combination mode over the Internet

Time(ms) / Number of Nodes	20	40	60	80
<i>Receiving query mean time (emulator)</i>	7539.92	13548.82	15575.06	21280.23
<i>Finding broker mean time (emulator)</i>	3271.46	11686.93	13613.34	13197.20
<i>Receiving query mean time (mobile)</i>	11876.76	14599.56	14238.75	14387.89
<i>Finding broker mean time (mobile)</i>	3285.76	5719.37	5975.78	5891.31

With a similar scenario, all virtual devices send their queries to one mobile anonymizer node. Table 6 shows the mean of the querying time in this scenario.

Table 6: Sending all load on a single mobile node over the Internet

Maximum wait time(ms)/number of emulator nodes	20	50	100
10000	4754.27	20948.73	55975.71
20000	4084.68	15154.70	42778.03
30000	4648.85	19952.28	46668.33
40000	4484.59	18305.65	48671.62
50000	5096.97	14300.29	43559.16
60000	4830.29	12913.29	52732.65

All of these experiments indicate that the  $P^4QS$  does not have a significant delay time or computational overhead. Thus, it can be easily deployed to achieve privacy with anonymization without any extra costs such as a trusted server.

## 6 Conclusion

In this paper, we introduced a novel peer-to-peer architecture for anonymizing location-based queries. In the proposed architecture, each node was responsible for anonymizing a zone and at the same time, it could use location-based services as a client peer. By using appropriate symmetric and asymmetric encryptions, each node just has the required information. So it could not have any more details about the owner of the query or the sequence of her queries. The ticket exchange protocol supported the execution and protected against DoS attacks. The  $P^4QS$  was implemented and tested on the local network and on the Internet. Experimental results showed that the  $P^4QS$  had a negligible effect on the time and computation costs.

## References

1. Hazim Almuhammedi, Florian Schaub, Norman Sadeh, Idris Adjerid, Alessandro Acquisti, Joshua Gluck, Lorrie Faith Cranor, and Yuvraj Agarwal. Your Location has been Shared 5,398 Times!: A field Study on Mobile APP Privacy Nudging. In *Proceedings of the 33rd Annual ACM Conference on Human Factors in Computing Systems*, pages 787–796. ACM, 2015.
2. Serge Gutwirth. *Privacy and the information age*. Rowman & Littlefield Publishers, 2002.
3. Urs Hengartner. Hiding location information from location-based services. In *Mobile Data Management, 2007 International Conference on*, pages 268–272. IEEE, 2007.
4. Miao Lin and Wen-Jing Hsu. Mining gps data for mobility patterns: A survey. *Pervasive and Mobile Computing*, 12:1–16, 2014.
5. Marco Gruteser and Dirk Grunwald. Anonymous usage of location-based services through spatial and temporal cloaking. In *Proceedings of the 1st international conference on Mobile systems, applications and services*, pages 31–42. ACM, 2003.
6. Bidi Ying and Dimitrios Makrakis. Protecting location privacy in vehicular networks against location-based attacks. *International Journal of Parallel, Emergent and Distributed Systems*, 30(2):101–117, 2015.
7. Tun-Hao You, Wen-Chih Peng, and Wang-Chien Lee. Protecting moving trajectories with dummies. In *Mobile Data Management, 2007 International Conference on*, pages 278–282. IEEE, 2007.
8. Baik Hoh and Marco Gruteser. Protecting location privacy through path confusion. In *Security and Privacy for Emerging Areas in Communications Networks, 2005. SecureComm 2005. First International Conference on*, pages 194–205. IEEE, 2005.
9. Panos Kalnis, Gabriel Ghinita, Kyriakos Mouratidis, and Dimitris Papadias. Preventing location-based identity inference in anonymous spatial queries. *Knowledge and Data Engineering, IEEE Transactions on*, 19(12):1719–1733, 2007.
10. Bugra Gedik and Ling Liu. Protecting location privacy with personalized k-anonymity: Architecture and algorithms. *Mobile Computing, IEEE Transactions on*, 7(1):1–18, 2008.
11. Mohamed F Mokbel, Chi-Yin Chow, and Walid G Aref. The new casper: query processing for location services without compromising privacy. In *Proceedings of the 32nd international conference on Very large data bases*, pages 763–774. VLDB Endowment, 2006.
12. Andreas Pfitzmann and Marit Hansen. A terminology for talking about privacy by data minimization: Anonymity, unlinkability, undetectability, unobservability, pseudonymity, and identity management, 2010.
13. Alastair R Beresford and Frank Stajano. Location privacy in pervasive computing. *Pervasive Computing, IEEE*, 2(1):46–55, 2003.



14. Rongxing Lu, Xiaodong Li, Tom H Luan, Xiaohui Liang, and Xuemin Shen. Pseudonym changing at social spots: An effective strategy for location privacy in vanets. *Vehicular Technology, IEEE Transactions on*, 61(1):86–96, 2012.
15. Alastair R Beresford and Frank Stajano. Mix zones: User privacy in location-aware services. In *Pervasive Computing and Communications Workshops, IEEE International Conference on*, pages 127–127. IEEE Computer Society, 2004.
16. Tanusri Bhattacharya, Lars Kulik, and James Bailey. Automatically recognizing places of interest from unreliable gps data using spatio-temporal density estimation and line intersections. *Pervasive and Mobile Computing*, 19:86–107, 2015.
17. Walid G Aref and Hanan Samet. Efficient processing of window queries in the pyramid data structure. In *Proceedings of the ninth ACM SIGACT-SIGMOD-SIGART symposium on Principles of database systems*, pages 265–272. ACM, 1990.
18. Joseph Meyerowitz and Romit Roy Choudhury. Hiding stars with fireworks: location privacy through camouflage. In *Proceedings of the 15th annual international conference on Mobile computing and networking*, pages 345–356. ACM, 2009.
19. Roman Schlegel, Chi-Yin Chow, Qiong Huang, and Duncan S Wong. User-defined privacy grid system for continuous location-based services. *Mobile Computing, IEEE Transactions on*, 14(10):2158–2172, 2015.
20. Meysam Ghaffari and Nasser Ghadiri. Ambiguity-driven fuzzy c-means clustering: How to detect uncertain clustered records. *arXiv preprint arXiv:1409.2821*, 2014.
21. Aniket Pingley, Wei Yu, Nan Zhang, Xinwen Fu, and Wei Zhao. Cap: A context-aware privacy protection system for location-based services. In *Distributed Computing Systems, 2009. ICDCS'09. 29th IEEE International Conference on*, pages 49–57. IEEE, 2009.
22. Kang G Shin, Xiaoen Ju, Zhigang Chen, and Xin Hu. Privacy protection for users of location-based services. *Wireless Communications, IEEE*, 19(1):30–39, 2012.
23. Haibo Hu and Jianliang Xu. Non-exposure location anonymity. In *Data Engineering, 2009. ICDE'09. IEEE 25th International Conference on*, pages 1120–1131. IEEE, 2009.
24. Zhigang Chen. *Energy-efficient information collection and dissemination in wireless sensor networks*. PhD thesis, The University of Michigan, 2009.
25. Liz Ribe-Baumann and Kai-Uwe Sattler. A hierarchical approach to resource awareness in dhts for mobile data management. *Pervasive and Mobile Computing*, 15:113–127, 2014.
26. Ion Stoica, Robert Morris, David Karger, M Frans Kaashoek, and Hari Balakrishnan. Chord: A scalable peer-to-peer lookup service for internet applications. *ACM SIGCOMM Computer Communication Review*, 31(4):149–160, 2001.
27. Gildas Avoine, Muhammed Ali Bingol, Xavier Carpent, and Siddika Berna Ors Yalcin. Privacy-friendly authentication in rfid systems: On sublinear protocols based on symmetric-key cryptography. *Mobile Computing, IEEE Transactions on*, 12(10):2037–2049, 2013.
28. Chi Wang, Hua Liu, Kwame-Lante Wright, Bhaskar Krishnamachari, and Murali Annavaram. A privacy mechanism for mobile-based urban traffic monitoring. *Pervasive and Mobile Computing*, 20:1–12, 2015.
29. Jiadi Yu, Hongzi Zhu, Haofu Han, Yingying Jennifer Chen, Jie Yang, Yanmin Zhu, Zhongyang Chen, Guangtao Xue, and Minglu Li. Senspeed: Sensing driving conditions to estimate vehicle speed in urban environments. *Mobile Computing, IEEE Transactions on*, 15(1):202–216, 2016.